PostBL: Post-Mesh Boundary Layer Mesh Generation Tool

Rajeev Jain and Timothy J. Tautges

Argonne National Laboratory, Argonne, IL 60439, USA

Abstract. A boundary layer mesh is a mesh with dense element distribution in the normal direction along specific boundaries. PostBL is a utility to generate boundary layer elements on an existing mesh model. PostBL supports creation of hexahedral, prism, quad, and tri boundary layer elements. It is formulated as an algorithm in MeshKit, which is an open-source library for mesh generation functionalities. Typically, boundary layer mesh generation is a premeshing process; in this effort, however, we start from a model that has already been meshed. Boundary layer elements can be generated along the entire skin or on selected exterior or internal surface boundaries. PostBL mesh operation can be coupled with other MeshKit meshing operations such as Jaal, NetGen, TetGen, and CAMAL and custom meshing tools such as RGG. Simple examples demonstrating generation of boundary layers on different mesh types and the OECD Vattenfall T-Junction benchmark hexahedral mesh are presented.

1 Introduction

Boundary layers are typically used for CFD applications, in regions with strong gradients-turbulent flow, diffusion-type equations, laminar flow or no-slip boundary with strong gradient of velocity [23]. Most other tools for generating boundary layers insert them before bulk mesh generation; in this effort, however, we develop a tool called PostBL and start from an already meshed model. This strategy is particularly useful when the meshing of the original problem is complicated and involves geometry decomposition and other geometry modifications prior to meshing. A utility for adding layers of elements along the exterior or interior (shared by two materials) boundary is developed. It relies on the edge or surface (2D or 3D) mesh on which the boundary layer elements are desired. This method is also useful for creating boundary layer elements in the wake regions along the direction of fluid flow. After boundary layer insertion, smoothing can optionally be used to improve mesh quality. Typically in external aerodynamic simulations, the type of turbulent modeling decides the thickness and number of boundary layer elements; PostBL would save time and eliminate the complexity involved in remeshing the entire model.

Most boundary layer meshing tools and algorithms treat boundary layer generation as a premeshing operation. This strategy works well for tri, quadrilateral, and tetrahedral meshes, since robust automatic mesh generation for these mesh types is available. However, for complex hexahedral meshes that are usually a combination of mapping, sweeping, and other such techniques, preboundary layer methodology is difficult to achieve in cases where decomposition boundaries cross the boundary layers.

Refining techniques stand in the middle of premesh and postmesh boundary layer methods, since material and boundary condition definitions are applied after completion of the refining operation. Also, refining is often not applicable to decomposition surfaces intersected by boundary layers, a combination of s urfaces, or complex thin boundary models. Tools such as CUBIT [25] have problems during the refining operation after geometry-based meshing has been done. The reason is that placement of boundary layer nodes on interior surfaces disrupts the usual body-associated mesh characteristics of the model. Several issues encountered during preboundary layer generation are also applicable to postboundary layer tools; Section 2 identifies various contributions and work been done in this area. Section 3 discusses MeshKit and other libraries used for developing this algorithm. Section 4 describes the PostBL algorithm. Section 5 presents examples highlighting the capabilities of PostBL, including results with the T-junction benchmark problem. Section 6 presents a brief summary and future work, and Section 7 gives concluding remarks.

2 Background

Many researchers have worked in the field of performance and generation of boundary layer elements for CFD simulations. Application-specific as well as generic tools and algorithms have been developed for generating boundary layers. While significant contributions have been made in the field of tetrahedral mesh generation and hexahedral mesh refinement for boundary layer generation, relatively little work has been done in postmesh boundary layer generation. Some commercial and academic tools have capabilities similar to PostBL, but they are either proprietary or not easily available. Karman [9] presented a linear-elastic smoothing scheme to push bulk mesh and generate a new, unstructured viscous layer of elements. However, he doesn't mention the ability to handle multiple materials and normal computation strategy when multiple materials are encountered on the boundary regions. Guillaume et al [7] in their research note mention an improvement to Karman's [8] approach in handling sharp corners; the main focus of this work is on on tetrahedral meshes. Ito and Nakahashi [8] proposed a postmeshing technique to create hybrid boundary layer elements; their work is also limited to tetrahedral, prism and pyramid elements. Botasso and Detomi [2] also present a postmesh boundary layer technique, which is again limited to tetrahedral meshes. A mesh motion algorithm is initially used to deflate the original mesh and make room for the boundary layer mesh. A stack of prisms is than inserted and tetrahedronized. Their algorithm depends on a prism-splitting template, calibration, and user tuning. PostBL is more general and can be applied to a variety of mesh types.

Most tetrahedral boundary layer mesh generation techniques initially fill up the boundary layer region and then mesh the bulk region. Some methods are based on directional grid refinement procedures for accurate solution of the boundary layer and wake flow regions. Bahrainian and Mehrdoost [1] describe an automatic tetrahedral mesh generator that captures the boundary layer and wake flow region. They compare the turbulent compressible flow solutions with published data indicating better results. Pirzadeh [20] also describes a new bounding-box-based meshing technique to generate high-quality tetrahedral meshes with boundary layers for aerospace and other applications; he also created VGRIDns, a tetrahedral grid generator developed at NASA Langley Research Center. Garimella and Shephard [5] present a generalized advancing layers method that can be used for non manifold models. They discuss basic technical formulations for boundary layer mesh generation, multiple growth curves at boundary layer nodes, node placement, spacing types, and checks for element validity at crossover including smoothing, shrinking, and deletion operations to avoid element collision. The focus of their work is on tetrahedral meshes; decomposed surfaces and hex-based multimaterial models are not discussed. Loseille and Löhner [11] study anisotropic mesh adaptation with boundary layer mesh generation. Highlights of generating tetrahedral meshes for automatic adaptive procedures for RANS on complex geometries are presented. Initially, they start from specifying boundary layers as a global mesh generation procedure; later they report problems with the approach and start "local recovery procedures" which is the most robust of the three “methods” mentioned by the authors. In another paper [12], they discuss various aspects of the point insertion methodology, including optimization, use of normal, and revaluation after each layer insertion. All of the above work can be classified as pre-boundary layer mesh generation.

Several publications report on generating boundary layer elements in hexahedral meshes. Quadros and Shimada [22] discuss a method for hex mesh generation of thin-section solids. Zhang et al [33] describe an octree-based isoconturing method for automatic multimaterial tetrahedral and hexahedral mesh generation. Merkley et al [14] present methods and applications of sheet insertion in a hexahedral mesh; their basic idea relies on the dual of quad or hex (2D or 3D, respectively). Boundary layer creation by insertion of layers of sheets of hexes along the boundary is described. Wang and di Mare [32], in their paper about automatic hex mesh generation for turbo-machinery applications, describe a technique to create boundary layer elements by carefully placing, maintaining, and dicing a buffer layer around a geometry. Their work also preserves material and boundary conditions after the boundary layer mesh generation process; they make use of specific templates for boundary layer insertion, and their method is closely tied to automation and turbomachinery applications. In another recent study, Maréchal [13] presents an improvement on octree-based methods for handling sharp features and generating boundary layers.

In this paper we address the problem of generating boundary layers elements for geometry decomposition boundaries that are typically observed during hex meshing across the boundary layer region. Boundary layer generation for multi-material models and wake regions are supported for hex, tet, quad, and tri meshes. Multiple surfaces can be specified for boundary layer generation. PostBL caters to the need for generating boundary layer elements for reactor assembly and core meshes. Given an element budget, mesh generation using paving or other such techniques for a large number of rods drilled in a reactor model is difficult. With PostBL layers of elements can be added to an already set-up coarse-meshed reactor core model. Specifying boundary layers as new materials can generate concentric instrumentation pins or narrow gap regions in reactor cores, which are otherwise hard to generate. PostBL has been implemented by using the MeshKit design philosophy; interoperability and use of various libraries to implement the algorithm and packages highlighted here will benefit coupling and extending this work for related tasks.

3 MeshKit and Implementation

PostBL relies on geometry and mesh libraries developed as part of the Interoperable Tools for Advanced Petascale Simulations (ITAPS) project. The common geometry module (CGM) [27] provides functions for constructing, modifying, and querying geometric models in solid model-based and other formats. CGM can evaluate geometry from several underlying geometry engines. ACIS [26] and Open Cascade [19] based geometries are used in this work. Finite-element mesh and mesh-related data are stored in the Mesh-Oriented datABase (MOAB) [28], which provides query, construction, and modification of finite-element meshes, plus polygons and polyhedra. CGM and MOAB can be accessed through the ITAPS iGeom and iMesh interfaces [18], respectively.

The iMesh concept of sets and tags is important to the implementation of these tools. A set is an arbitrary collection of entities and other sets; a tag is data annotating entities, sets, or the interface. The combination of sets and tags is a powerful mechanism used to describe boundary conditions, material types, and other types of metadata commonly found with mesh.

MeshKit [15] is an open-source mesh library under development at Argonne National Laboratory. It is targeted to researchers, tool developers, and users wishing to generate meshes. MeshKit approaches geometry-based meshing and other mesh operations as a digraph-based process. This graph functionality is imported from the Lemon graph library [4]. This approach uses a two-phase graph execution: setup and execution. In the setup phase, algorithms express their requirements; automatic sub algorithmic node creation takes place in this phase. In the execution phase, algorithms actually get executed. Mesh algorithms are represented as graph nodes, and graph edges are dependencies between algorithms. External packages such as CAMAL [3], NetGen [16], TetGen [24], MESQUITE [10], and Ipopt [31] are implemented as graph algorithms for various mesh-related tasks. Meshing algorithms can be applied to a geometric entity or a collection of entities; operations can also be based on the entities created as a result of the previous operation. The graph-based approach allows explicit representation of meshing dependencies, which may allow parallelization of the meshing process. This approach also facilitates updating the mesh after small changes. In Fig. 1 a digraph for meshing a cylinder geometry is shown. First the surface is quad-meshed by using a quadmesh operation. Then a sweep operation fills the cylinder with hexes. Both quadmesh and sweep are user-specified mesh operations; MeshKit creates automatic graph nodes-IntervalMatch, EdgeMesh, and MapMesh-during the setup phase of the quadmesh and sweep algorithms. In Fig. 1, S1, S2, and S3 are the surfaces; C1 and C2 are the curves; and V1 is the volume forming the cylinder. The type of geometric entity that each mesh operation works on is highlighted inside its respective parallelogram.

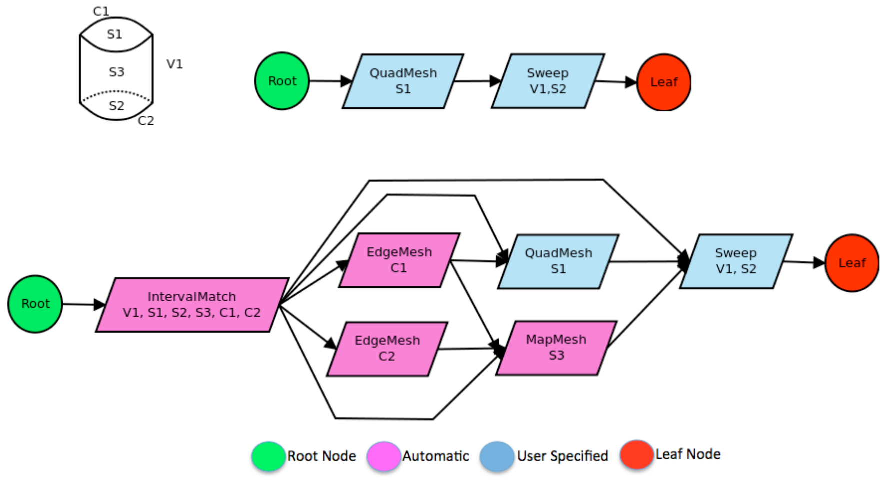

Fig. 1. Digraph-based mesh generation example: sweep-meshing a cylinder.

The PostBL mesh operation adheres to this digraph-based approach. PostBL can directly read a mesh file, acting as a stand-alone operation, or get a mesh from the previous graph node. A python interface to MeshKit algorithms helps in scripting meshing problems. All meshing algorithms including PostBL can be accessed through MeshKit's python interface based on PyTAPS [21]. Figure 2 shows a users-specified digraph that generates a reactor assembly with a boundary layer mesh from scratch. The AssyGen [29] operation generates geometry from the text-based input file describing a reactor assembly. This geometry is input to the Jaal [30] mesh operation, which feeds into the Extrude mesh operation to generate a 3D mesh. Then PostBL, based on the user-specified inputs, generates the desired boundary layer elements in the model.

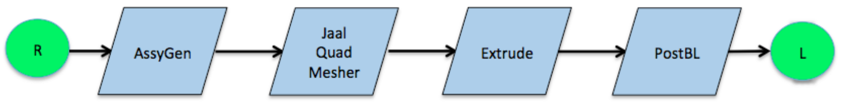

Fig. 2. User-specified digraph for creating reactor assembly mesh with boundary layers.

4 Algorithm

PostBL is a simple mesh operation implemented as one of the digraph-based algorithms in MeshKit. PostBL mesh operation can be run as a C++ MeshKit executable or as a python task using MeshKit's python interface. All mesh file format supported by MOAB [28] can be used as an input/output file. The core algorithm is as follows.

1. Determine the mesh dimension, and collect the boundary elements and nodes. Establish material definition for new elements. If no specific material id is specified, new elements are added to original model elements from which boundary layer elements are derived.
2. For all material nodes on boundary, tag nodes that are a part of more than one material. Also, find out the number of material boundaries occurring at each node.
3. Check for material interface elements that have only an edge or node (2D or 3D) on the boundary. Nodes of such elements are tagged to indicate multiple normal vectors during node creation.
4. Set the sizes of adjacency and connectivity arrays based on nodal tags in steps 2 and 3 and the type of the newly created elements.
5. Loop through all the boundary nodes.
    a. Find adjacent entities of the same and higher dimension for normal computation.
    b. For nodes with multiple materials, obtain geometric boundary direction for generating boundary layer nodes. If this node has a material partially connected, also create an average normal from adjacent material boundaries.
    c. Compute normal direction for node creation.
    d. For the specified number of intervals, create boundary layer nodes based on the specified thickness and bias.
6. Reset the coordinates of the bulk mesh with the coordinates of the innermost boundary layer nodes created in 5d.
7. To detect collision, check for the Jacobian of boundary elements of the bulk mesh. If collision is detected, attempt to locally modify the penultimate elements for creating positive Jacobian.
8. Loop through the specified boundary layers (edges for 2D, surfaces for 3D).
    a. Create boundary layer elements from nodes created in 5d.
    b. Associate new elements with geometry (if available).
    c. Set material and boundary sets for newly created elements.
9. Set a "fixed" tag for an optional mesh smoothing operation after the PostBL operation.
10. Report clock and CPU time, and save the final mesh.

4.1 Boundary Layer Specification

Basic specification parameters such as thickness and number of intervals along the boundary are assumed to remain constant throughout the model. A bias variable indicating the ratio of subsequent node spacing toward the boundary can be specified. Bias between 0 and 1 indicates finer elements along the boundary. Bias greater than 1 indicates finer elements toward the bulk mesh. Negative bias is invalid. The boundary layer can be specified as a surface/edge id (3D or 2D) or as a collection in the form of a NEUMANN set. For wake regions, the side that stays fixed must be specified. By default newly created elements are added to their corresponding parent material sets; alternatively, newly created elements can be added to a material set defined as input (see Section 5.3). Appendix A shows a sample keyword-based input file to the program. All these keywords can also be specified as a variable in python script.

4.2 Collision Detection

Postmesh boundary layer generation algorithms do not usually encounter the problem of collision with bulk mesh. This situation is true for most hex-meshed CFD models, since only a fraction of edge length is typically used for boundary layer generation. In PostBL, after pushing the bulk mesh for boundary layer creation, the deformed elements are checked for positive Jacobian. If negative Jacobian is detected, local modifications are made to coordinates of the element and its neighboring elements to generate positive Jacobian. The algorithm stops after a predefined number of attempts for local modification of elements.

4.3 Normal Computation and Handling of Multiple Materials

Several special cases are encountered during normal computation, which generally is the average normal of all the adjacent boundary layer entities. When handling multiple materials along the boundary, material/volume modification during the boundary layer generation process must be avoided to prevent any change in material volume during the process. Special tags are created on boundary layer nodes to indicate material boundaries. When a node that is a part of multiple materials is encountered; a normal is created along each material boundary to maintain constant material volume. Also, for material interface elements that have only an edge or node (2D or 3D) on the boundary, an additional normal is created to produce valid and better quality elements. Figure 3(a) and Fig. 3(c) show two initial mesh models with multiple materials on the boundary. Boundary layers are desired on the bottom edge of this 2D quad mesh. At the node where materials meet, only one normal along material boundaries is required for Fig. 3(a), while for mesh in Fig. 3(c) three normal directions are required for creating a valid mesh. In general, models with sharp corners and edges may require multiple normal directions for node creation, but this is not done in the work here. It is assumed that sharp corners and edges are adequately meshed for postmesh operation to generate good quality elements.

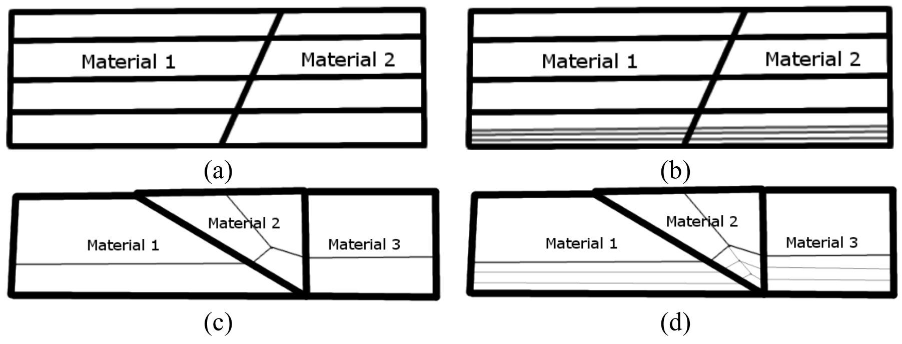

Fig. 3. Handling multiple materials along boundary layers: (a) Original quad mesh with 2 materials. (b) Final mesh after generating boundary layers. (c) Original quad mesh with 3 materials. (d) Final mesh after generating boundary layers.

4.4 Smoothing

After the PostBL mesh generation is completed, smoothing can optionally be applied to improve mesh quality. Smoothing can be local or global based on which entities get specified for smoothing. With local smoothing a fixed tag is applied on all nodes except nodes near the boundary layer region. Local refining may help improving mesh quality and/or obtain a gradual change in mesh size from boundary layers to bulk mesh. Fixed nodes must be carefully chosen to preserve the boundary layer region. Smoothing the boundary layer nodes may result in loss of bias in the boundary elements created by PostBL. With global smoothing nodes in material boundaries are fixed, and the smoothing algorithm can change all interior nodes. An example of local smoothing is given in Section 5.1.

5 Results

We present simple and complex examples both carefully chosen to demonstrate the features and limitation of the PostBL tool.

5.1 Simple Tri Mesh

Figure 4(a) shows a simple tri mesh. The orientation of the tri elements on the boundary layer elements can be slanting upwards or downwards based on the simulation requirements. Figures 4(b) and 4(d) show the output from PostBL. TriScheme variable (see Appendix A) of 0 or 1 can be specified for generating these results. These meshes contain large-aspect-ratio/low-shape-quality triangles in the penultimate layer of the boundary layer triangles. Figure 4(e) and 4 (f) shows the improvement in the shape quality metric of the resulting mesh before and after using Mesquite. Only the bulk mesh is smoothened in this case, although based on the requirements either/both bulk and boundary layer mesh could be smoothened. When boundary layers collide with bulk mesh, new elements are still created and could be restored during the smoothing phase.

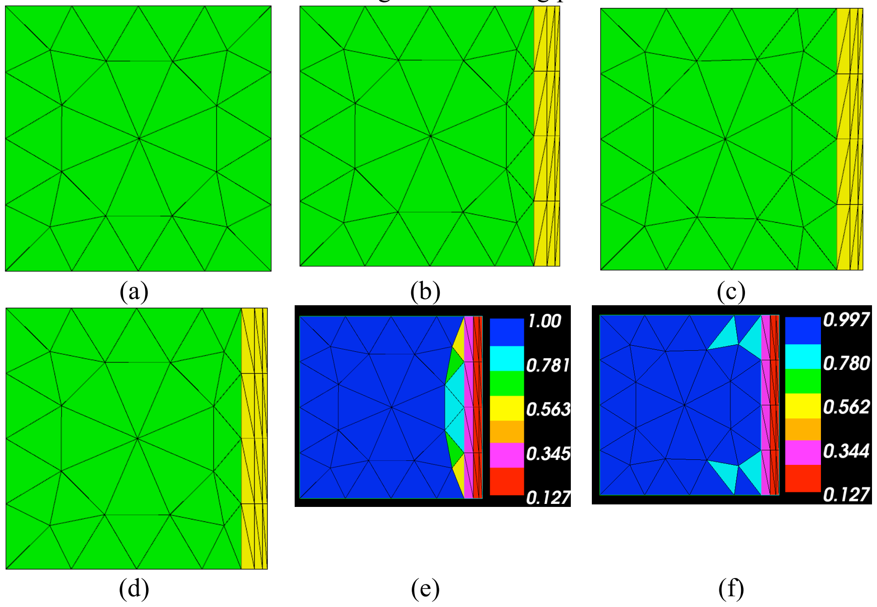

Fig. 4. Tri mesh example: (a) Original mesh. (b) Three boundary layers on right (slanting downwards). (c) Bulk mesh smoothened using Mesquite. (d) Three boundary layers (slanting upwards). (e) Shape quality metric of mesh shown in Fig. 4(d) and 4(f). (f) Shape quality metric after smoothing bulk mesh.

5.2 Exterior Boundary Layers on a Quad Mesh

Figure 5 shows a simple plate with an elliptical cavity and its corresponding boundary layer elements. At the two corners of the cavity the quality of the elements is poor. Several boundary layer algorithms create a multiple normal at points where the normal from neighboring elements differs by a certain percentage. Since this is a postmesh operation, the original mesh is responsible for capturing the boundary features of the model. In this algorithm the average normal of all the boundary layer elements is chosen for creation of new boundary layer elements. The choice of creating only one normal from each boundary layer node is reasonable for meshes that have the boundary layer surfaces sufficiently resolved. We note that the use of tuck or wedge-type node placement would better account for the local curvature observed here.

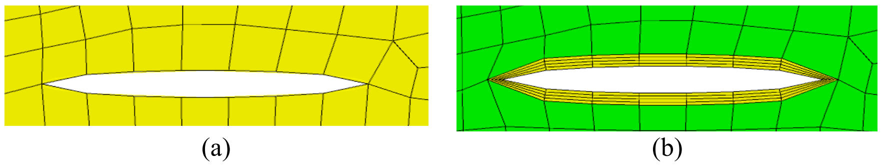

Fig. 5. Quad mesh example: (a) Original mesh. (b) Mesh with 4 boundary layers.

5.3 Hex-Meshed Reactor Assembly

The next example demonstrates internal boundary layers on a hex mesh, since elements are present on both sides of the boundary. Figure 6(a) shows a hex-meshed nuclear reactor assembly containing six fuel rods and one control rod at the center. The solid line on the control rod indicates the original geometric body. Layers of boundary elements can be generated and optionally set-up as a new material type. Setting boundary layers as new material can be useful for creating instrumentation pins and fluid gaps concentric to the original rods. Several types of reactor assemblies have a lot of fuel rods drilled on them, and it is hard to pave the surface of the geometry; complexity increases as the number of concentric rods to the fuel rods increases. Also, because of the high temperature of the fuel rod, heat flux near the fuel rods is high, thereby requiring a finer mesh in this region. In Fig. 6, side surfaces of all cylindrical rods are set-up as input boundary layers. Three types of boundary layer meshes can be generated as shown in Fig. 6(b), 6(c) and 6(d). Based on the requirements of fluid dynamics, neutronics, or structural mechanics varying combinations on interior or exterior boundary layers can be generated. In Fig. 6(b), the boundary layer elements are generated toward the cylindrical region, whereas in Fig. 6(c) the boundary layer elements are generated away from the cylindrical region. Using the FIXMAT keyword can specify bulk mesh that does not change during the boundary layer creation. When creating elements on both sides of the boundary, different thickness and bias can be specified for each side of the boundary. A bias of 0.7 is used for this example. Fig. 6(d) shows boundary layer elements generated on both sides of the cylindrical region. This feature is useful for creating boundary layer meshes in wake regions where the boundary layer is surrounded on both sides by materials.

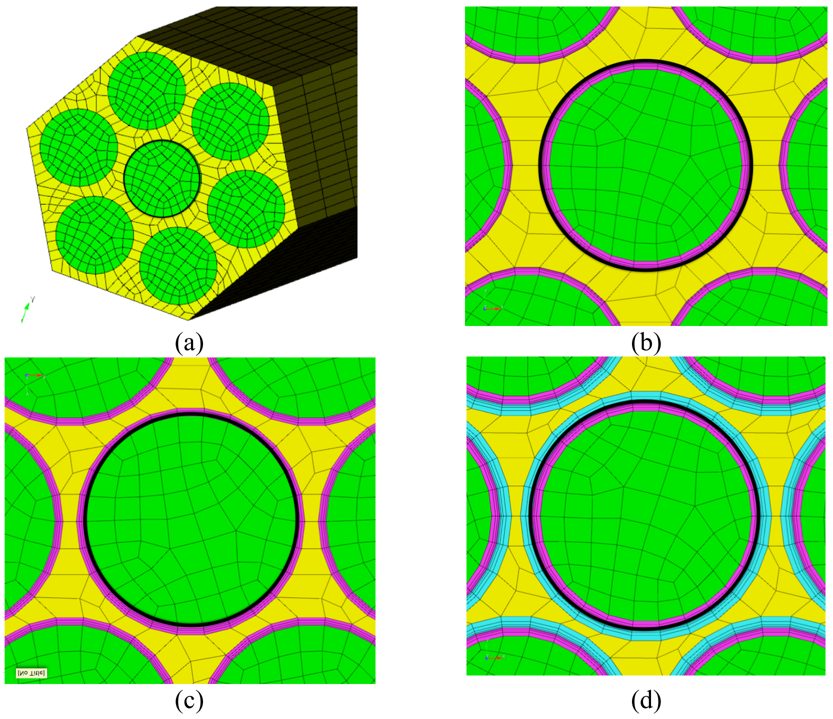

Fig. 6. Quad mesh example with elements on both sides: (a) Original mesh. (b) Mesh with 4 boundary layers toward the cylinders (c) Mesh with 4 boundary layers away from the cylinders. (d) Boundary layers on both sides of the cylinders.

5.4 Simple Tetrahedral Mesh

Figures 7(a) and 7(b) show 12 tetrahedral elements on a cube geometry; Figures 7(c) and 7(d) highlight the hybrid mesh (tetrahedral and prism elements) with 4 prism elements added to the existing mesh. Gmsh [5] is used to display the shrunken elements in this model.

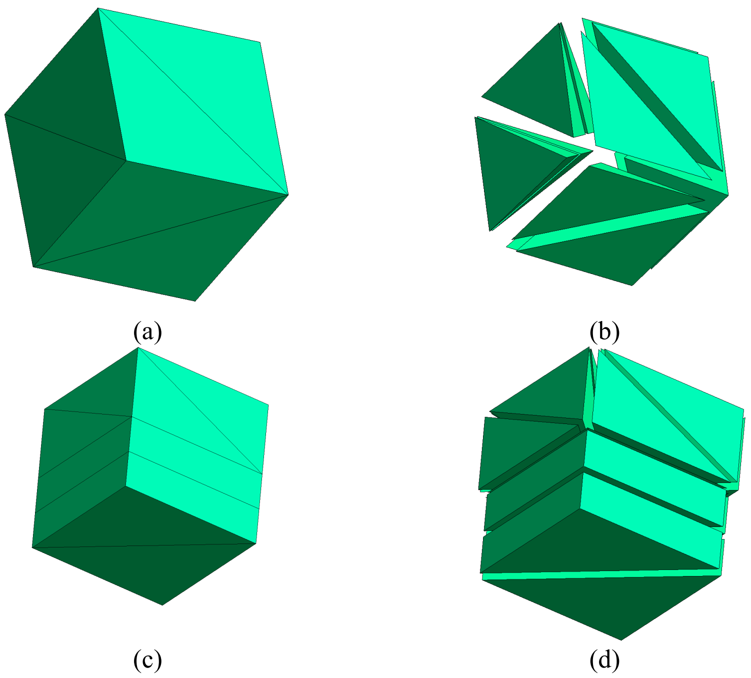

Fig. 7. Tet mesh example: (a) Original mesh. (b) Original mesh with elements shrunk by a factor of 80%. (c) Mesh with two boundary layer prisms. (d) Final mesh shrunk by 80% showing tetrahedral and prism elements.

5.5 19 Assembly Hex-Meshed Reactor Core

Figure 8(a) shows a reactor core created by using MeshKit/RGG tools: AssyGen and CoreGen [29]. Based on the user-specified description, AssyGen first creates the two types of reactor geometries involved in the model. The outer covering and the two assemblies are meshed separately by using MeshKit algorithm(s). CoreGen then is used to copy/move/merge the three mesh files and form the core mesh. Four layers of boundary layer elements are then added to the fluid region and gap region between the assemblies. Actual reactor core models have hundreds of assemblies forming the reactor; adding boundary layers during the generation of core mesh is difficult because of the complexity and the number of parts involved in the model. PostBL preserves the initial material and boundary conditions prescribed by RGG and allows for creation of varying boundary layers for different meshes used in a multiphysics simulation. Figure 8(b) shows a close-up view of the region highlighted in Fig. 8(a). Note that only the gap and fluid region are shown in Fig. 8(b). Figure 8(c) shows a close-up view of the mesh after the PostBL operation. In addition to the fuel and gap regions, fuel pins are shown in-order to clearly identify the change from the initial mesh.

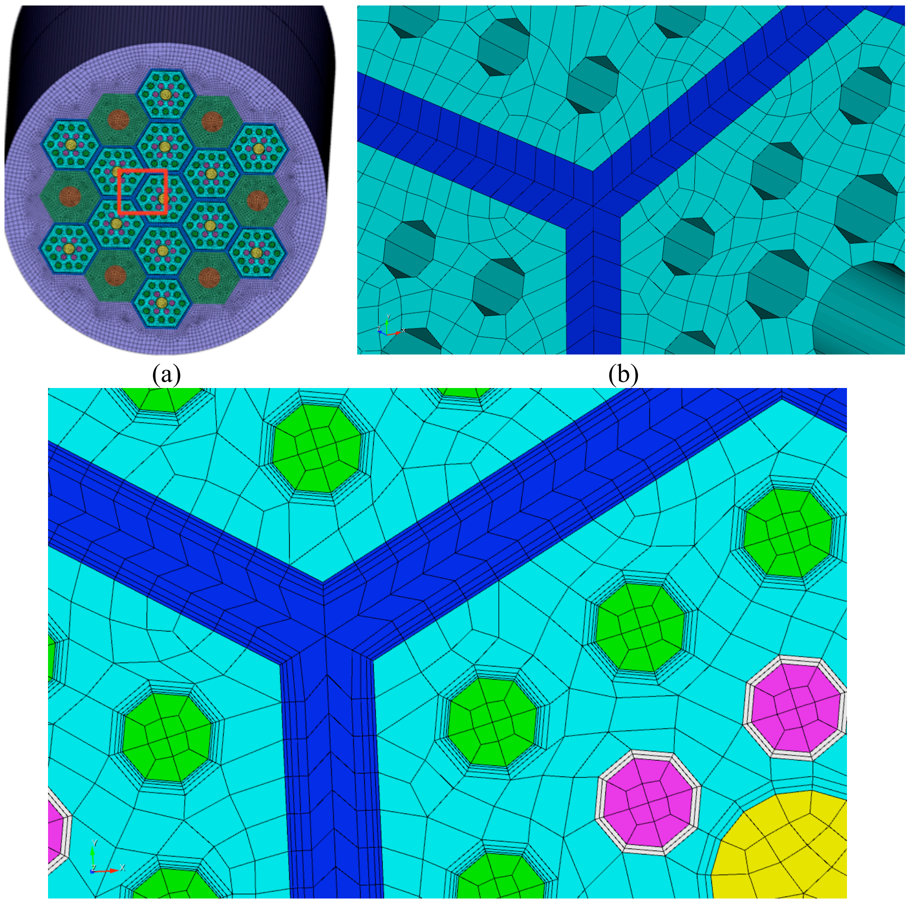

Fig. 8. 19 assembly reactor core mesh: (a) Original mesh. (b) Close-up of original mesh showing fluid and gap regions. (c) Close-up of original mesh showing boundary layers on fluid and gap regions.

5.6 Hex-Meshed T-Junction Benchmark Problem

Figure 9(a) shows the OECD Vattenfall T-Junction benchmark mesh that was used in a blind benchmark of various CFD codes. The Argonne code Nek5000 participated in this benchmark using this mesh [17]. The mesh was generated with CUBIT 10.2, then modified to extend the (top) inflow and (side) outflow pipes. Non circular surfaces are Neumann boundary layers for PostBL operation; six layers of boundary elements are added. Figure 9(b) shows a zoomed-in view of the T-junction in the original mesh. PostBL uses only one normal direction on this T-intersection point to create new boundary layer elements; this allows for creation of a smooth mesh with similar-sized elements. Creating multiple normal at sharp features such as this T-junction may result in better-quality elements, but this would be also cause the creation of very small elements that may not be desired by the simulations. The aspect ratio of hex elements of this mesh is shown in Fig. 9(c); elements with lower aspect ratio are on the boundary and acceptable. A close-up in Fig. 9(d) shows a zoomed view of the T-junction with six new layers as a separate material. Mesquite is used for improving the quality of hex elements in the bulk region that are close to the boundary layer elements. Figure 9(e) shows the zoomed view of the T-junction with improved shape of the hex near the boundary layers.

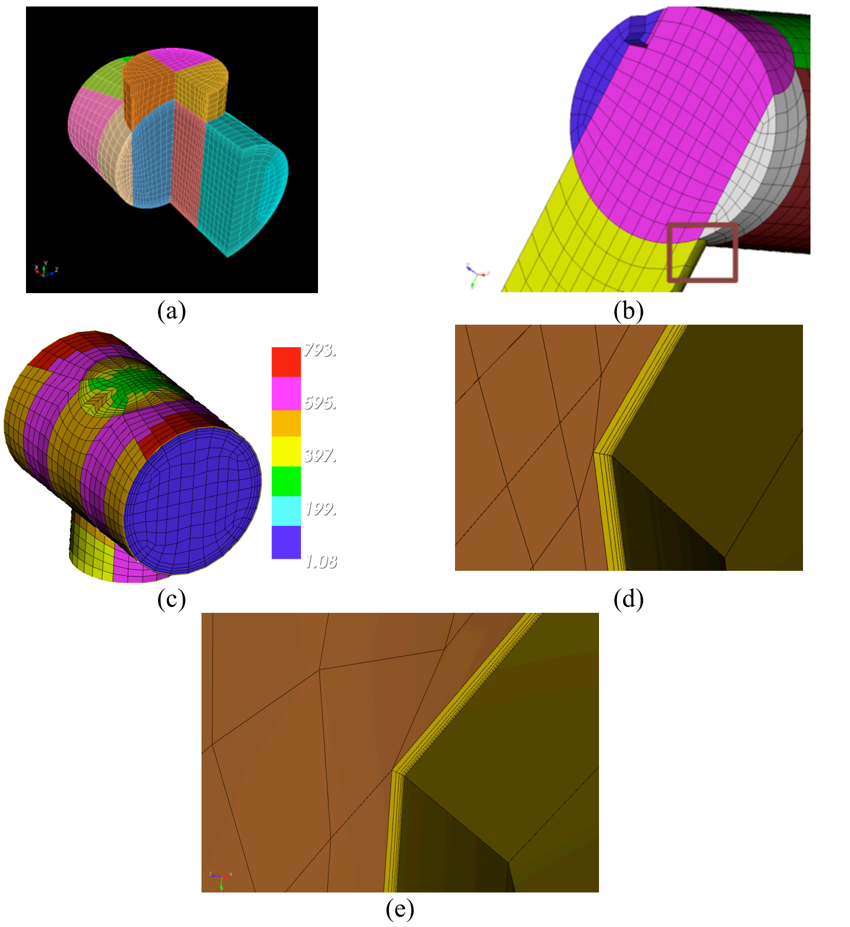

Fig. 9. Hex meshed T-junction example: (a) Sectional view of the original mesh. (b) Close-up sectional view highlighting the T-junction. (c) Mesh with 6 boundary layers showing aspect ratio of the final mesh. (d) Close-up along T-junction with 6 boundary layers generated by PostBL. (e) Close-up along T-junction after smoothing using Mesquite.

6 Discussion and Future Work

PostBL is useful for CFD analysts studying solution parameters by varying the number of boundary layers. Remeshing of the entire model is required if pre-boundary layer methods are utilized, a time consuming and unstable strategy for complicate problems. It is desired that the mesh input to PostBL resolve the geometry around the boundary. When there is no collision with other boundaries, high-quality boundary layers can be generated by using PostBL. The final mesh produced by PostBL can be smoothened to improve the mesh quality at the interface between the bulk and new boundary layer mesh. Generic templates for refinement of hex, tri, and quad meshes during the post mesh boundary layer operation can be added to provide more element choices for the users. In its current implementation PostBL is serial; more work is needed for distributed parallelization of the tool. Further research also can be done on utilizing the quality metrics of the modified bulk mesh for point insertion and normal calculation. MeshKit's graph-based meshing process is explained in this work, since PostBL can be used for boundary layer generation after meshing using MeshKit algorithms.

7 Conclusion

PostBL is well suited for complicated geometries that are difficult to mesh by using automatic mesh generation techniques. It can be applied to hex, tet, tri, and quad meshes. It presents an attractive alternative to conventional pre-boundary layer tools. PostBL can generate boundary layers on interior (shared by two materials) or exterior surfaces of an already existing mesh model. Material and boundary information about the initial model is preserved and extended to include boundary layer elements in the final model. PostBL is ideally suited for hexahedral meshes where decomposition boundaries intersect boundary layer regions. Several refinement tools such as CUBIT solely rely on the volume or surface of the model for boundary layer mesh generation, an approach that does not work with decomposed geometry. PostBL has been applied to reactor assembly and core meshes, and it can help increase the mesh resolution in coolant flow and wake regions.

APPENDIX

A. Sample .inp file specifying the parameters required by the PostBL tool.

```
! Name of the input mesh file
MeshFile 5b.cub

! Slant direction of tri-meshes 0 or 1
!TriScheme 0

! id of the neumann set on which boundary layer needs to be created.
NeumannSet 55
!Exlclaimation mark indicates comment line:
!To specify surface use either Surfaces or NeumannSet keyword.
! Id of the surface on which boundary layer needs to be created (commented).
!Surfaces 11

! Material id that will be assigned to newly created hexes.
Material 55

! Boundary layer thickness.
Thickness 0.3

! Number of layers.
Intervals 2

! Bias is the ratio of interval size of two consecutive boundary layers. >0.
Bias 1.0

! Name of output mesh file, can be any format that is supported by MOAB.
Outfile 5b_postbl.h5m

Debug 1
! This marks the end of input file for boundary layer generation.
END
```

Acknowledgments

We thank the Fathom group at Argonne, who maintain the libraries required by this tool. This work was supported in part by the U.S. Department of Energy Office of Nuclear Energy Nuclear Energy Advanced Modeling and Simulation (NEAMS) Program; by the U.S. Department of Energy Scientific Computing Research, Office of Science; and by the U.S. Department of Energy's Scientific Discovery through Advanced Computing program, under Contract DE-AC02-06CH11357.

References

1. Bahrainian, S. S., and Mehrdoost, Z. (2012). An automatic unstructured grid generation method for viscous flow simulations. *Mathematics and Computers in Simulation*.
2. Bottasso, C. L., and Detomi, D. (2002). A procedure for tetrahedral boundary layer mesh generation. *Engineering with Computers*, *18*(1), 66-79.

3. CAMAL - The CUBIT Adaptive Meshing Algorithm Library, Sandia National Laboratories, Albuquerque.
4. Dezs, B., Jüttner, A., & Kovács, P. (2011). LEMON - An open source C++ graph template library. *Electron. Notes Theor. Comput. Sci.*, *264*(5), 23–45.
5. Garimella, R. V., and Shephard, M. S. (2000). Boundary layer mesh generation for viscous flow simulations. *International Journal for Numerical Methods in Engineering*, *49*(1), 193-218.
6. Geuzaine, C., and Remacle, J. F. Gmsh: A three-dimensional finite element mesh generator with built-in pre-and post-processing facilities, Version 2.2. 4, 2008.
7. Guillaume, V., Fornier, Y., and Boubekeur, T. (2012). Hybrid Viscous Layer Insertion in a Tetrahedral Mesh, In *IMR* (Research Note).
8. Ito, Y., and Nakahashi, K. (2002, September). Unstructured Mesh Generation for Viscous Flow Computations. In *IMR* (pp. 367-377).
9. Karman, S. L. (2007). Unstructured viscous layer insertion using linear-elastic smoothing. *AIAA journal*, *45*(1), 168-180.
10. Knupp, P. (2006, September). Mesh quality improvement for SciDAC applications. In *Journal of Physics: Conference Series*, *Vol. 46*, No. 1.
11. Loseille, A., and Löhner, R. (2009). On 3D anisotropic local remeshing for surface, volume and boundary layers. In *Proceedings of the 18th International Meshing Roundtable* (pp. 611-630). Springer Berlin Heidelberg.
12. Loseille, A., and Löhner, R. (2013). Robust boundary layer mesh generation. In *Proceedings of the 21st International Meshing Roundtable* (pp. 493-511). Springer-Berlin.
13. Maréchal, L. (2009). Advances in octree-based all-hexahedral mesh generation: handling sharp features. In *Proceedings of the 18th International Meshing Roundtable* (pp. 65-84). Springer-Berlin.
14. Merkley, K., Ernst, C., Shepherd, J. F., and Borden, M. J. (2008, January). Methods and applications of generalized sheet insertion for hexahedral meshing. In *Proceedings of the 16th International Meshing Roundtable* (pp. 233-250). Springer-Berlin.
15. MeshKit: http://trac.mcs.anl.gov/projects/fathom/browser/MeshKit
16. NetGen – automatic mesh generator, Johannes Kepler University Linz, 2008.
17. Obabko, A. V., Fischer, P. F., Tautges, T. J., Karabasov, S., Goloviznin, V. M., Zaytsev, M. A., and Aksenova, A. E. (2011). *CFD validation in OECD/NEA t-junction benchmark* (No. ANL/NE-11/25). Argonne National Laboratory, Argonne, IL.
18. Ollivier-Gooch, C., Diachin, L., Shephard, M. S., Tautges, T., Kraftcheck, J., Leung, V., and Miller, M. (2010). An interoperable, data-structure-neutral component for mesh query and manipulation. *ACM Transactions on Mathematical Software (TOMS)*, *37*(3), 29.
19. Open CASCADE technology website (2000–2010) http://www.opencascade.org
20. Pirzadeh, S. Z. (2010). Advanced unstructured grid generation for complex aerodynamic applications. *AIAA Journal*, *48*(5), 904-915.
21. PyTAPS, website, https://pypi.python.org/pypi/PyTAPS/
22. Quadros, W. R., and Shimada, K. (2002). Hex-layer: layered all-hex mesh generation on thin section solids via chordal surface transformation. In *Proceedings of 11th Inter National Meshing Roundtable* (pp. 169-180)
23. Schlichting, H., and Gersten, K. (2000). *Boundary-Layer Theory*. Springer-Berlin.
24. Si, H. (2006, January). On refinement of constrained Delaunay tetrahedralizations. In *Proceedings of the 15th International Meshing Roundtable* (pp. 509-528). Springer Berlin Heidelberg.

25. Sjaardema, G.D, Tautges, T. J, Wilson, T. J, Owen, S. J, Blacker, T. D, Bohnhoff, W. J, Edwards, T. L, Hipp, J. R, Lober, R. R, and Mitchell, S. A (1994). CUBIT mesh generation environment, users manual, vol 1. Sandia National Laboratories, Albuquerque
26. Spatial website (2010) http://www.spatial.com/
27. Tautges, T. J (2005) CGM: a geometry interface for mesh generation, analysis and other applications. Eng Comput *17*:486–490
28. Tautges, T. J, Meyers, R, Merkley, K, Stimpson, C, Ernst, C (2004). MOAB: A mesh-oriented database, SAND2004-1592. Sandia National Laboratories, Albuquerque.
29. Tautges, T. J., and Jain, R. (2012). Creating geometry and mesh models for nuclear reactor core geometries using a lattice hierarchy-based approach. *Engineering with Computers*, *28*(4), 319-329.
30. Verma, C. S., and Tautges, T. (2012). Jaal: Engineering a high quality all-quadrilateral mesh generator. In *Proceedings of the 20th International Meshing Roundtable* (pp. 511-530). Springer-Berlin.
31. Wächter, A., and Biegler, L. T. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, *106*(1), 25-57.
32. Wang, F., and di Mare, L. (2013). Automated hex meshing for turbomachinery secondary air system. In *Proceedings of the 21st International Meshing Roundtable* (pp. 549-566). Springer-Berlin.
33. Zhang, Y., Hughes, T. J., and Bajaj, C. L. (2010). An automatic 3D mesh generation method for domains with multiple materials. *Computer Methods in Applied Mechanics and Engineering*, *199*(5), 405-415.